Informatics College Pokhara

Module Code & Module Title

CS4001 Programming

Assessment Weightage and Type

1st year Second semester

Student's Name London met ID

Anisha Gurung 24040771

Assignment Milestone Date:2025/3/9

Submitted to,

Sushil Paudel sir,

Sushil Parajuli sir,

Table of Contents

Table of Figures:

Tables:

Untitled document

Islington College,Nepal

Document Details

Submission ID
trn:oid:::3618:96196509

Submission Date
May 16, 2025, 12:35 PM GMT+5:45

Download Date
May 16, 2025, 12:36 PM GMT+5:45

File Name
Untitled document

File Size
24.1 KB

28 Pages

3,912 Words

19,613 Characters

12% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- 36 Not Cited or Quoted 11%
  Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%
  Matches that are still very similar to source material
- 3 Missing Citation 1%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

2% Internet sources
0% Publications
12% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

1.Introduction

As the current era is known as the era of digital and high-tech technology, it has shaped the way of our daily lifestyle and the things that we have to deal with in our daily life. The increase in dependency of digital platforms has created many golden opportunities for people and made life so easy. Just one click away from purchasing goods from anywhere in the corner and different digital platforms which are helpful for different levels of organization. This project aims to design and develop a gym membership application software.

This project combines creativity and logical thinking skills and for the user interface part we will implement the GUI. This course is designed to provide hands-on experience, emphasizing the practical use of JAVA programming language and the tool we use is BlueJ or IntelliJ idea which will make us familiar with code on different platforms.

This project helps to understand the deeper logic behind the application. The primary goal of this course is to build a user-friendly application for the gym members to track down their membership plan. It will provide necessary information such as name, age, email, their join date and attendance and many more.

1.1 Goals and objectives

The main goal of this project is to build a user-friendly gym membership application that provides and meets the needs of organizations. The goals and objectives are mentioned below:

Goals:

- Develop a user-friendly gym membership application
- Enhance the user-experience
- Taking the users' attendance to track down their attendance
- Keeping the track details of two different regular and premium members
- To make proper membership ship plan according to their budget

Objective:

- Building easy to use and easily understood buttons, text application
- Using the feature of Java language principles to manage gym memberships, payment and their active status and attendance.

2.Wireframes

Wireframe is a normal visual representation of how a project will look after the coding part. The process of designing a website or project at a structure level without proper functionality is called wireframing. Wireframes are used early in the development process to establish the basic structure of a page before visual design and content are added. Building a website is a process of making sure how a project will look and to make the website more user friendly. Wireframing is one of those parts of the web development process that should not be skipped, so that we can make the website in such level that it will be easily understandable by the users. At last, we can say that wireframes are essential to design a good mobile/web application. (GeeksforGeeks, 2024)

2.1 Wireframe of Gym membership application

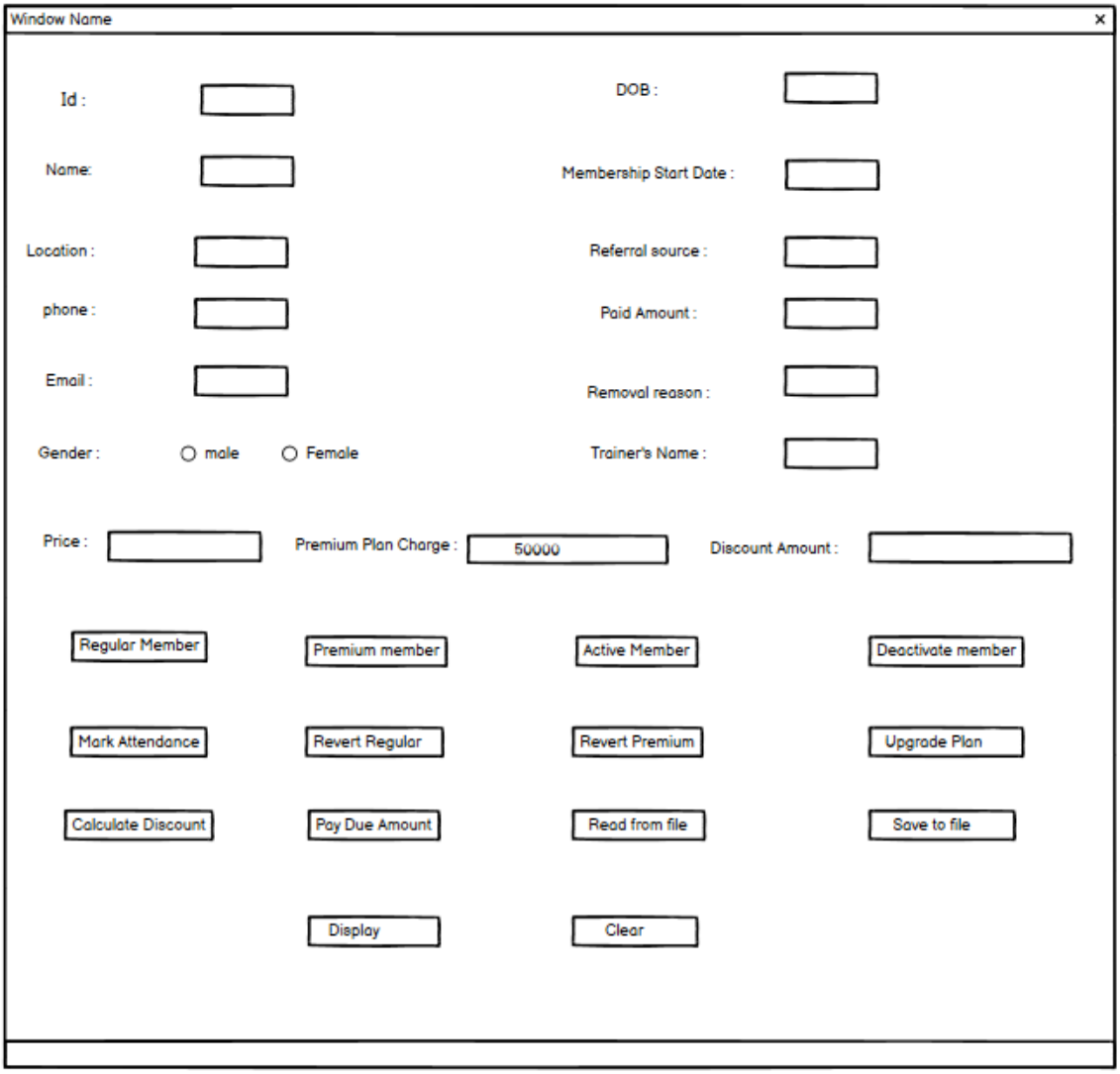

Figure 1:Wireframe of gym membership application

2.2 Screenshot of Developed GUI

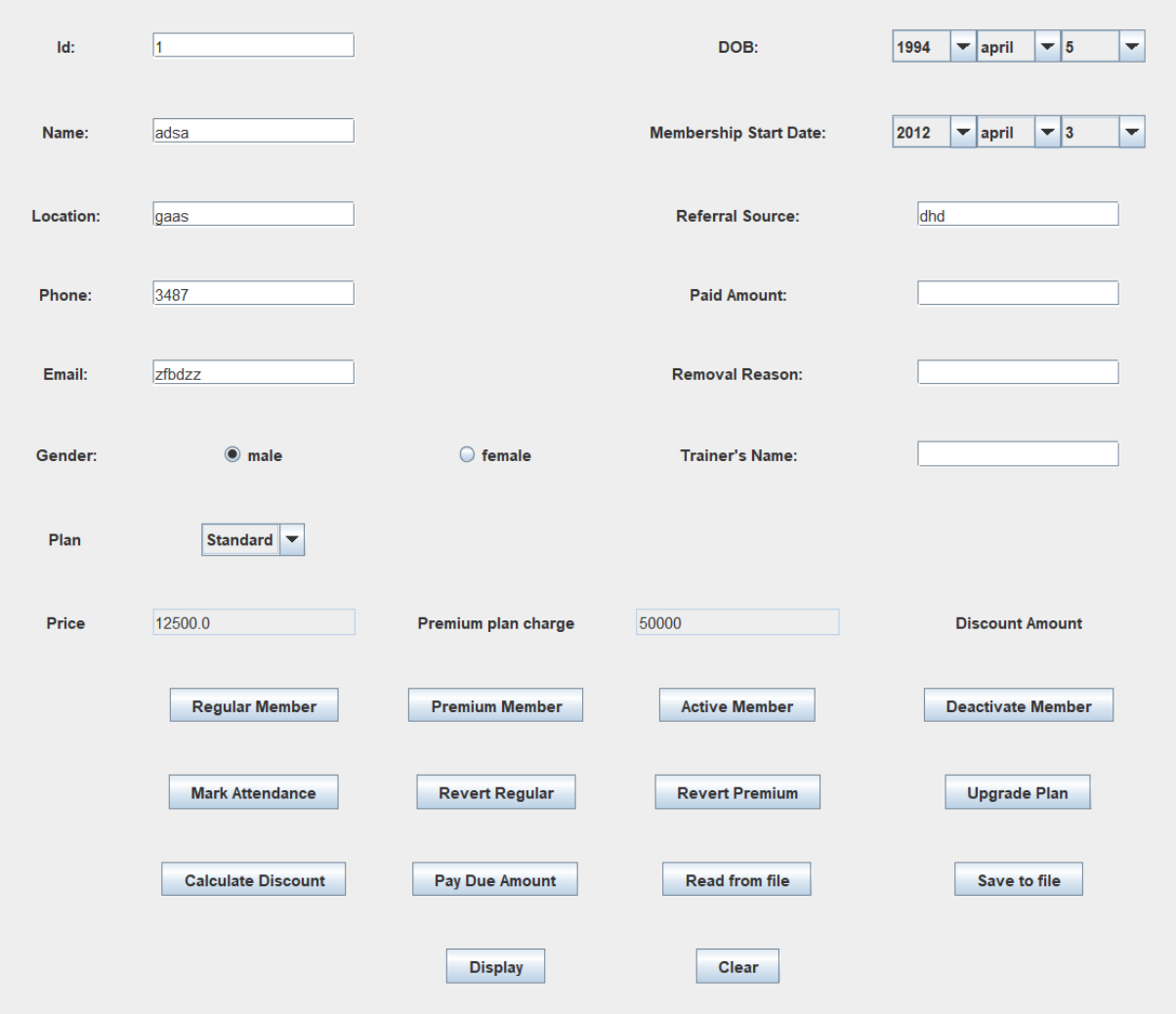

Figure 2:Screenshot of developed GUI

3.Class Diagram

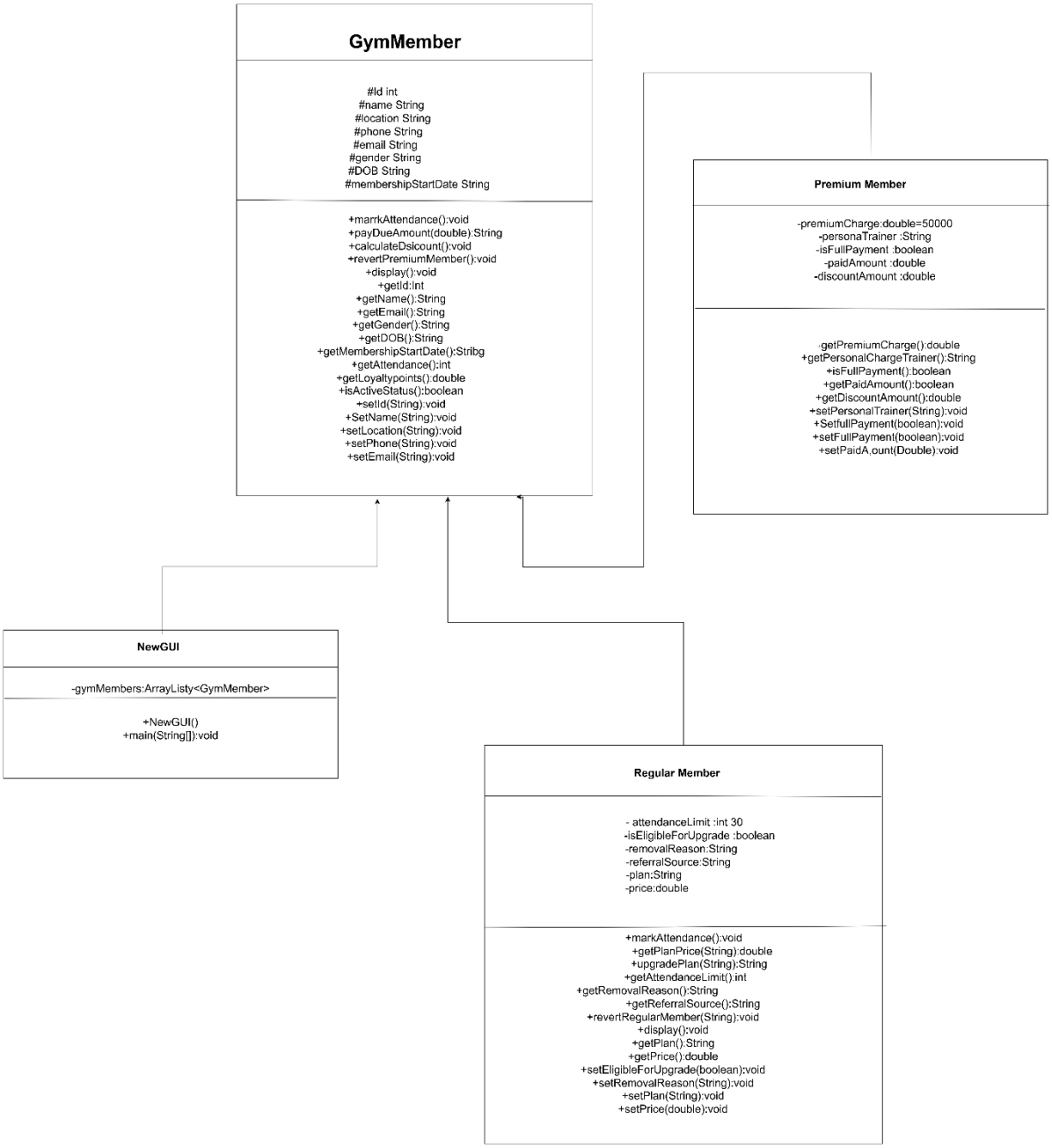

Figure 3:class diagram of Gym member

4.PseudoCode (Excluding setter and getter)

4.1PseudoCode for Gym membership

ABSTRACT CLASS GymMember

VARIABLES:

id - integer

name - string

location – string

phone - string

email - string

gender - string

DOB – string

membershipStartDate - string

attendance – integer

loyaltyPoints – double

activeStatus - boolean

CONSTRUCTOR GymMember(id, name, location, phone, email, gender, DOB, membershipStartDate)

id = passed id

name = passed

location = passed

phone = passed

email = passed

gender = passed

DOB = passed DOB

membershipStartDate = passed membershipStartDate

attendance = 0 loyaltyPoints = 0

```
activeStatus = false
ABSTRACT FUNCTION markAttendance()
 FUNCTION activateMembership()
      IF activeStatus is false THEN
           activeStatus = true
           PRINT "membership is active: " + name
          ENDIF
 FUNCTION deactivateMembership()
          IF activeStatus is true THEN
            activeStatus = false
            PRINT "membership is deactive: " + name
ENDIF
FUNCTION resetMember()
attendance = 0

loyaltyPoints = 0
PRINT "Reset membership: " + name
activeStatus = false
PRINT "Reset membership: " + name
 FUNCTION display()
       PRINT "id: " + id PRINT "name: " + name
     PRINT "location: " + location PRINT "phone: " + phone
     PRINT "email: " + email
     PRINT "gender: " + gender
     PRINT "DOB: " + DOB
    PRINT "joined date: " + membershipStartDate
```

PRINT "attendance: " + attendance

PRINT "loyalty points: " + loyaltyPoints

PRINT "active status: " + activeStatus

4.2 Pseudocode for PremiumMember

CLASS PremiumMember EXTENDS GymMember

VARIABLES:

premiumCharge = 50000

personalTrainer – string

isFullPayment – Boolean

paidAmount - double

discountAmount - double

CONSTRUCTOR PremiumMember(id, name, location, phone, email, gender, DOB, membershipStartDate, personalTrainer)

CALL super(id, name, location, phone, email, gender, DOB, membershipStartDate)
personalTrainer = passed

personalTrainer isFullPayment = false

paidAmount = 0

discountAmount = 0

FUNCTION markAttendance()

attendance = attendance + 1

loyaltyPoints = loyaltyPoints + 5

FUNCTION payDueAmount(amount)

IF isFullPayment is true THEN

RETURN "payment is already done"

ELSE

paidAmount = paidAmount + amount

IF paidAmount > premiumCharge THEN

```
                RETURN "payment is excess, please check"
        ELSE
            remainingAmount = premiumCharge – paidAmount
            IF paidAmount equals premiumCharge THEN
                isFullPayment = true
                RETURN "Payment successful! Premium charge " + premiumCharge + " paid"
            ELSE
                RETURN "Payment successful! Remaining: " + remainingAmount
            ENDIF
    ENDIF
ENDIF
 FUNCTION calculateDiscount()
IF isFullPayment is true THEN
 discountAmount = 0.1 * premiumCharge
 PRINT "Discount of " + discountAmount + " applied"
ELSE
       discountAmount = 0
       PRINT "No discount - payment not complete"
 ENDIF
FUNCTION revertPremiumMember()
 CALL super.resetMember()
personalTrainer = "" isFullPayment = false
 paidAmount = 0
discountAmount = 0
FUNCTION display()
CALL super.display()
```

```
PRINT "Personal Trainer: " + personalTrainer
PRINT "Paid Amount: " + paidAmount
PRINT "Full Payment: " + isFullPayment
 remainingAmount = premiumCharge - paidAmount
IF isFullPayment is false THEN
        PRINT "Remaining Amount: " + remainingAmount
ENDIF
 IF isFullPayment is true THEN
          PRINT "Discount Amount: " + discountAmount
 ENDIF
```

4.3 Pseudocode for RegularMember

```
CLASS RegularMember EXTENDS GymMember
VARIABLES:
          attendanceLimit = 30
          isEligibleForUpgrade – Boolean
        removalReason - string
         referralSource – string
      plan – string
  price - double
CONSTRUCTOR RegularMember(id, name, location, phone, email, gender, DOB, membershipStartDate, referralSource)
CALL super(id, name, location, phone, email, gender, DOB, membershipStartDate)
referralSource = passed referralSource
isEligibleForUpgrade = false
 plan = "basic"
 price = 6500
removalReason = ""
```

```
FUNCTION markAttendance()
        attendance = attendance + 1
        loyaltyPoints = loyaltyPoints + 5
    IF attendance >= attendanceLimit THEN
        isEligibleForUpgrade = true
    ENDIF
FUNCTION getPlanPrice(plan)
    IF plan equals "basic" THEN
        RETURN 6500
ELSE IF plan equals "standard" THEN
        RETURN 12500
ELSE IF plan equals "deluxe" THEN
        RETURN 18500
ELSE RETURN -1
ENDIF
FUNCTION upgradePlan(newPlan)
IF isEligibleForUpgrade is false THEN
    RETURN "Not eligible for upgrade yet"
ELSE
    newPrice = getPlanPrice(newPlan)
    IF newPrice equals -1 THEN
        RETURN "Invalid plan"
ELSE
        IF plan equals newPlan THEN
            RETURN "Already on " + newPlan + " plan"
        ELSE
```

```
                    plan = newPlan
                      price = newPrice
                      isEligibleForUpgrade = false
RETURN "Plan upgraded to " + newPlan + " for " + price
                ENDIF
            ENDIF
 ENDIF
 FUNCTION revertRegularMember(removalReason)
          CALL super.resetMember()
         isEligibleForUpgrade = false
         plan = "basic"
          price = 6500
         removalReason = passed
         removalReason
FUNCTION display()
         CALL super.display()
        PRINT "Plan: " + plan
         PRINT "Price: " + price
 IF removalReason not equals "" THEN
            PRINT "Removal Reason: " + removalReason
ENDIF
```

5.Method Description

5.1: Description of initialize Components method

All of the GUI components are set up in this process. By connecting each text field, label, and button to its associated instance variable, it creates instances of each of these elements. A new DesignTextField is used for each text field, a new

DesignLabel is used for each label, and a new DesignButton is used for each button. This process is typically used to ensure that all components are initialized appropriately during GUI setup

5.2: Description of newGUI Constructor

This newGUI class is a graphical user interface where all the necessary buttons ,labels,textfields and methods are available and also calling the necessary method from other classes also. In this class newGUI we have instance variable,arraylist and also constructor . An ArrayList named gymmembers is decalerd to stored all the details of members in the gym.

- Add Regular button
  When this button is clicked after entering the necessary text fields then ,if the id is valid then the member is saved as regular member in gym members.
- Add Premium button
  When this button is clicked after entering the necessary text fiels then, if the id is valid then the member is saved as premium member in gym members.

- Active Member
  We must first enter all required information in order for this active button to function. After verifying that the ID is valid, it will call the active member method from the GymMember class, which requires that the active status be initially fase. Only then can we activate the member, as there will be a logical error if the button is already activating the members.

- Deactivate Member
  Similar to active member, this button requires that we first enter all required information. It will then verify that the ID is valid and call the active member method from the GymMember class, which has the condition that the deactivate status be set to true because only then can we set it back to false; otherwise, a logical error will also occur.

- Mark attendance

After completing the necessary information and activating the premium ID, this button is clicked. When the attendance is entered, the loyalty points also increase by five points. For regular members, the situation is slightly different; the mark attendance is called, and the loyalty points also increase by five points. Additionally, if a regular member wishes to upgrade their plan, their attendance must be greater than or equal to the attendance limit, which is thirty, before they are deemed eligible to do so.

- Revert Regular
  When this button is clicked it call the method revert regular from regular member class and it calls the super member restMember and eligible for upgrade ,plan,price and removal reason are again reset

- Revert Premium
  When this button is clicked it call the revertPremuimMember from premium class where the method is said that it will call the restMember from super class and set other variable such as personal trainer name ,full payment and paid amount and discount amount are reset again

- Upgrade button
  When this upgrade button is clicked it should call the upgradePlan method and in the method it is said that if the eligible is false then print the message saying not eligible and if the price is -1 then the invalid plan will show an another message saying "Invalid plan. Please select a valid plan."
  Or else show message plan successfully upgraded.

- Calculate button
  When this button is clicked the calculateDiscount method is called and then the condition there is goes like this is payment if full then only the discount is given to that amount for premium charge which is 10 % only here the total

charge is 50000 and is payement is full then due to 10 % discount we get 5000 discount amount.

- Pay due Amount
  When this button is clicked it calls the method pay due amount which will say that if the payment is full return message your payment is done and else if you have not paid full but only half then the remaining amount will show in message box and later when you pay the remaining amount of the same id It will add that remaining amount and the previous amount and if the payment is excess then then it will show a message box saying your payment is excees .Please check again thank you.

- Read from file
  This button when clicked this will creates a new window titled “Member Details” and set a text area which cannot be edited and the memberdetails.txt file will open and reads the file each line by line and shows the content in the text area .

- Save to file
  When this button is clicked it checks if there are any gym members to save and if none shows then an error message is shows it creates a file name memberDetails.txt and checks if there are any members or not if there are then shows their details and shows an error message when done

- Display
  When this button is clicked it checks if there are any members to display and if non shows a error message and creates a separate window and here shows the window with all members details.
- Clear
  When this button is called then it will call the clearButton method which will resets all the necessary textfields .

5.3 Description of main method

This main method calls the constructor’s name NewGUI

6. Testing

6.1. Test 1: Adding a regular member

Table 1:Table of Test 1

Objective	To add a regular member
Action	fill up all the necessary text field and click the regular button in the GUI
Expected Output	The regular gym member should be added successfully with an appropriate message dialog box.
Actual Output	The regular gym member added successfully with an appropriate message dialog box.
Result	The test was successful.

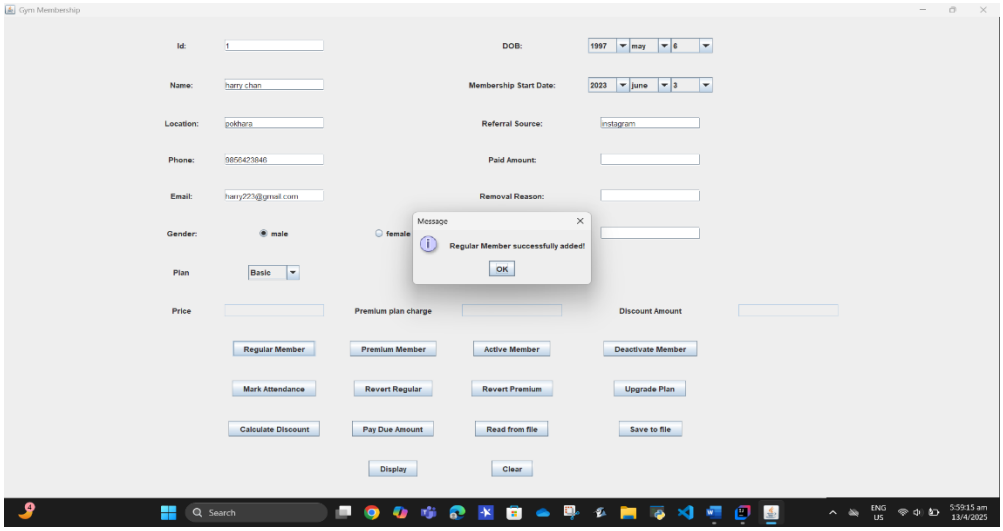

Figure 4:Screenshot of adding regular member

6.2. Test 2: Adding a premium member

Table 2:Table of Test 2

Objective	To add a premium member
Action	Fill up the necessary text filed and click on the premium button to add the member
Expected output	The premium gym member should be added successfully with an appropriate message dialog box.
Actual output	The premium gym member added successfully with an appropriate message dialog box.
Result	The test was successful.

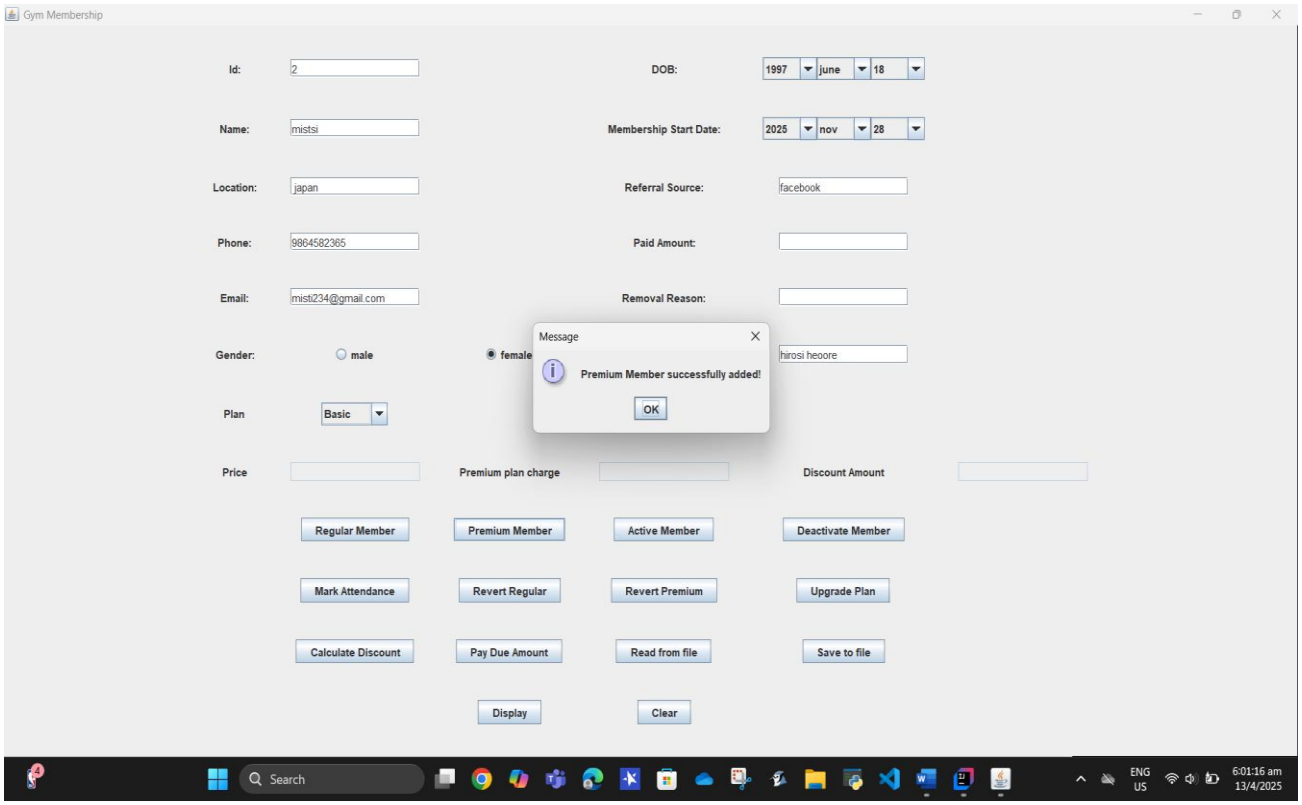

Figure 5:screenshot of adding premium member

6.3. Test 3: Compile and run using command prompt

Table 3:Table of Test 3

Objective	Compile and run the program using command prompt

Action	Go to the file where the program is, in the search bar type cmd and terminal will open then you should type java the name of your Gui class.java and again type java and the name of your Gui class
Expected output	The output the gui class will open
Actual output	The output was opened
Result	The test was successful.

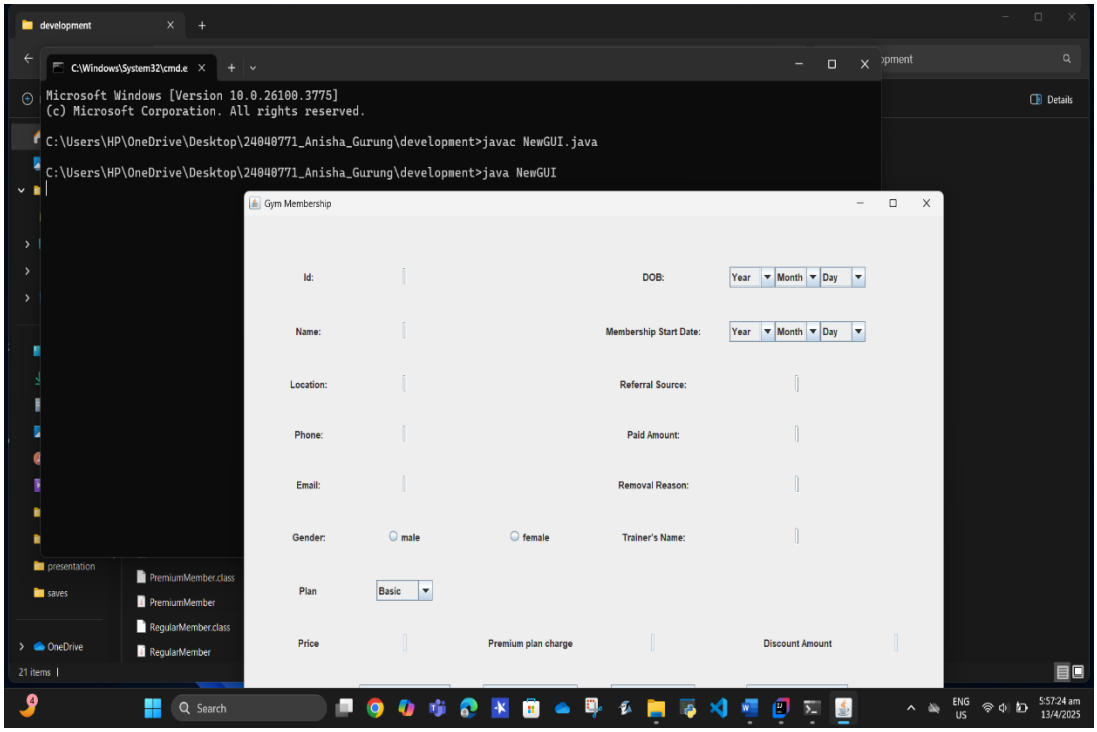

Figure 6:Screenshot of compile and running program using command prompt

6.4. Test 4: Test Case for Mark attendance

Table 4:Table of Test 4

Objective	Test case for mark attendance

Action	After you have added regular member or premium member you can enter the id and, now click on mark attendance button
Expected output	After clicking now, it should show the attendance with an appropriate message dialog box.
Actual output	It showed the dialog box with attendance and loyalty counts
Result	The test is successful.

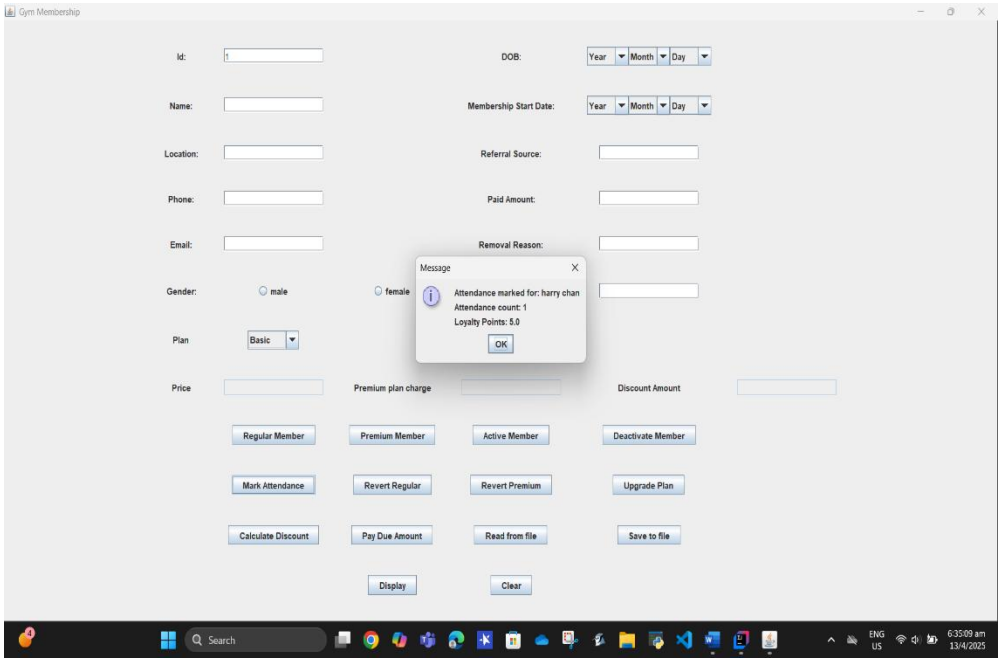

Figure 7:screenshot of test of mark attendance

6.5. Test 5: Test for upgrade plan

Table 5:Table of Test 5

Objective	To upgrade plan of regular member to standard or deluxe

Action	To upgrade plan first you need to enter the id and the id need to be active and the attendance score must be above 30 or equal to 30 then only eligible for upgrade
Expected output	Upgrade is done with a price updated according to the plan selected and a pop-up message showing saying plan successfully upgraded
Actual output	A pop-up message with upgraded plan and price according to the plan
Result	Test successful

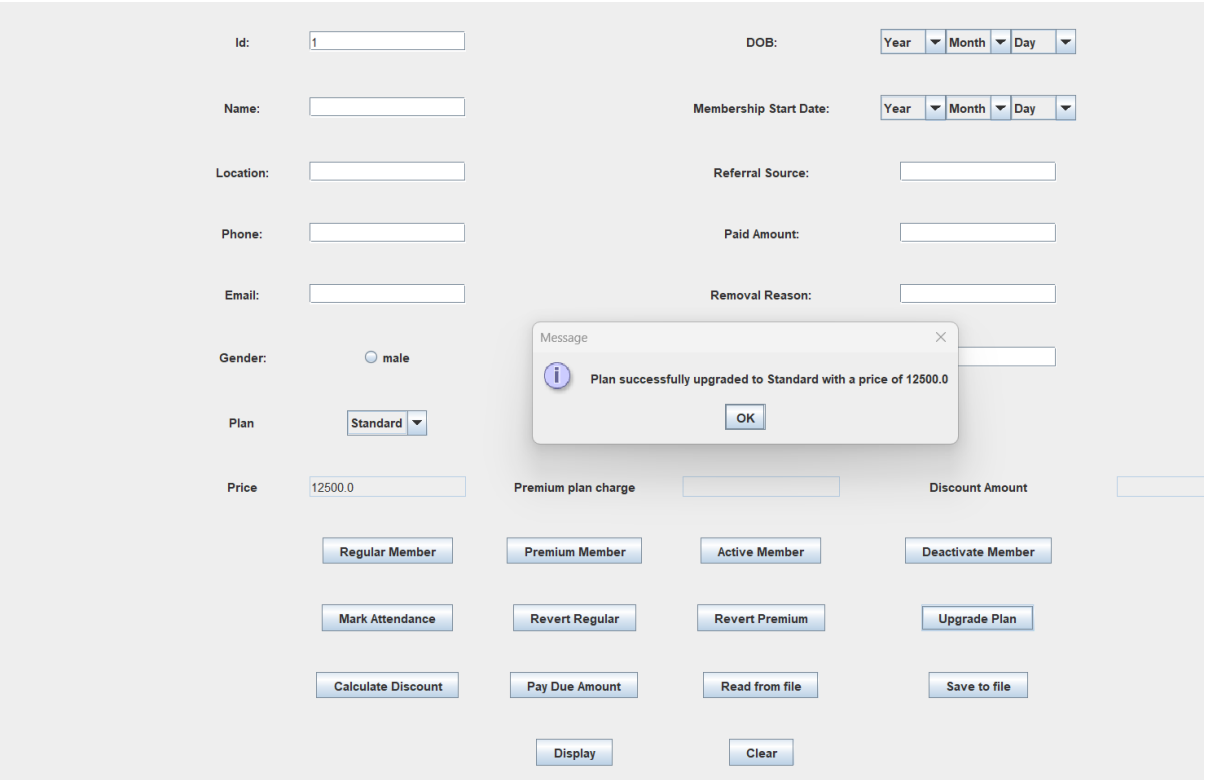

Figure 8:screenshot of upgraded plan

6.6. Test 6: Test for Pay due Amount

Table 6:Table of Test 6

Objective	To pay the amount of premium charge
Action	First need to active the premium id and then need to pay the amount since the total charge is 50000 need to enter the amount
Expected output	If the amount is full then message popup with successful if the amount is less then total charge, then popup message should show remaining amount
Actual output	Shows the payment successful
Result	Test successful

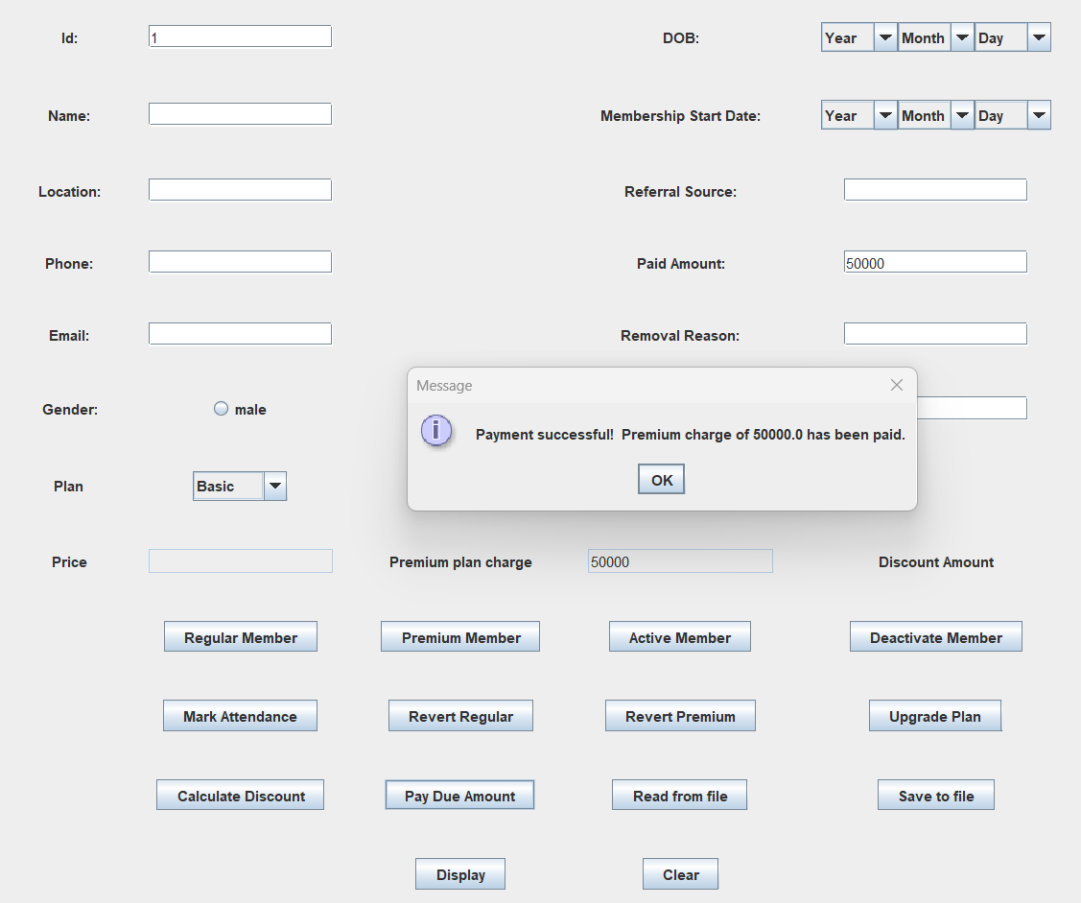

Figure 9:screenshot of pay due amount

6.7. Test 7 Test for Revert Regular member

Table 7:Table of Test 7

Objective	To revert the regular member
Action	Enter the valid id if id exist in the system, and click on the revert regular button
Expected output	Shows a pop-up message after clicking the button
Actual output	It showed the pop-up message
Result	Test successful

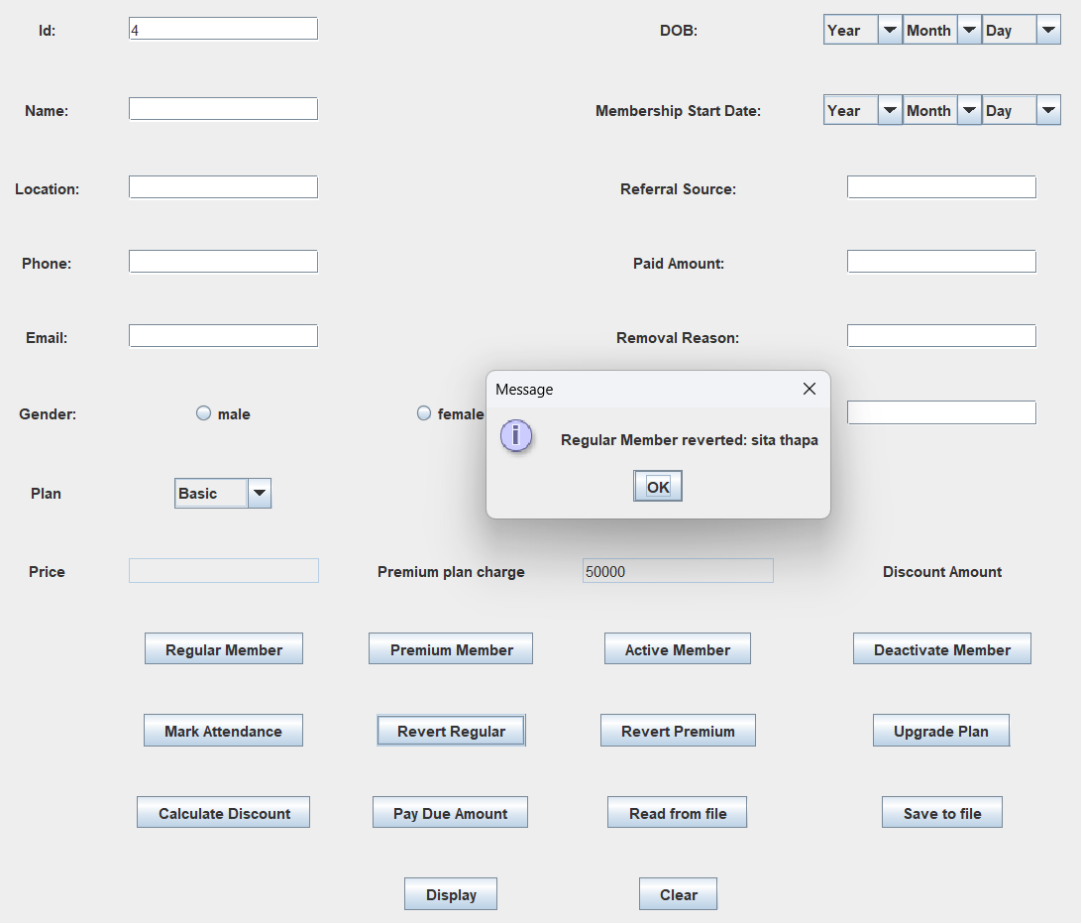

Figure 10:screenshot of for revert regular member

6.8. Test 8: Test for Revert premium member

Table 8:Table of Test 8

Objective	To revert the premium member
Action	Enter the valid id if id exist in the system, and click on the revert premium button
Expected output	Show a pop-up message after clicking the button
Actual output	It showed the pop-up message
Result	Test successful

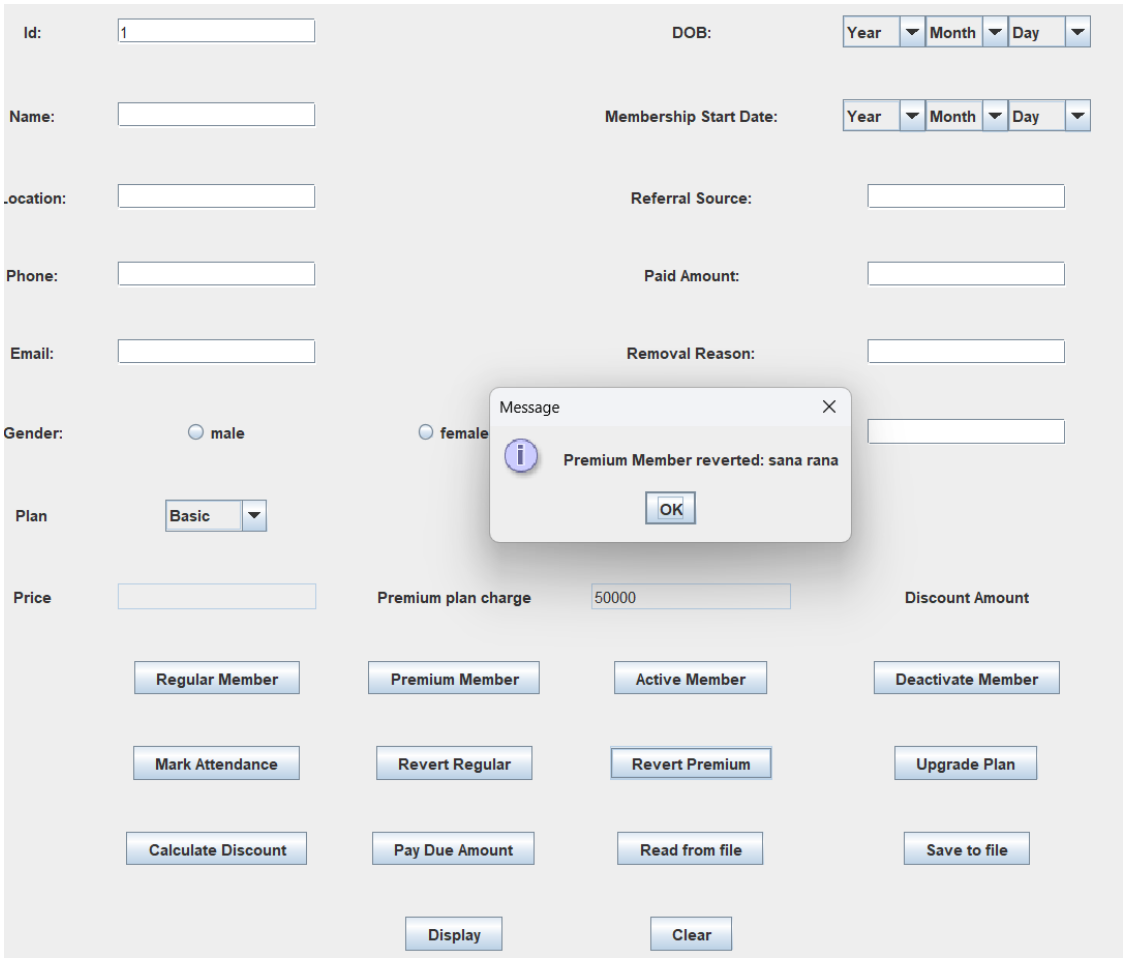

Figure 11:screenshot of premium member

6.8. Test 8: Test for save to file

Table 9:Table of Test 9

Objective	To save the file in a memberDetails.txt file in the program about the details entered
Action	First, enter all the necessary text filed according to the member regular and premium and then click the save button
Expected output	Should save the details in txt file and a message pop up should show confirming that is saved
Actual output	Shows a pop-up message and saves it in the txt file.
Result	Test successful.

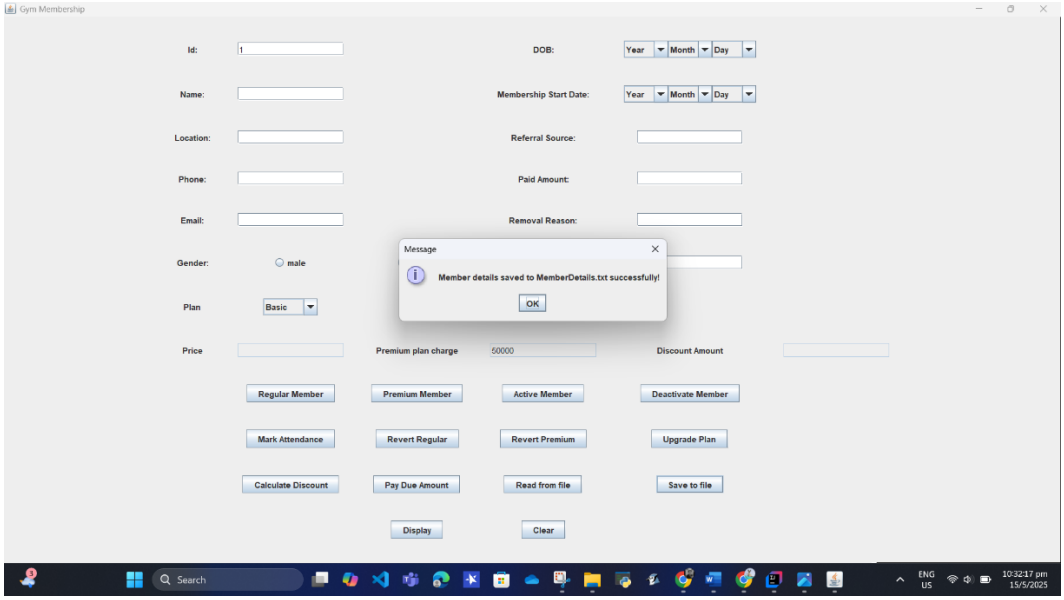

Figure 12:screenshot of member saved successfully

6.9. Test 9: Read from file

Table 10:Table of Test 10

Objective	To read the file from memberDetails.txt where save files are saved and show the read details in a new frame
Action	Enter the id of the member whose details need to be read and click on the read button

Expected output	Open a new frame where details are also displayed
Actual output	A new frame is opened and the details which were in the details.txt file are displayed
Result	Test successful

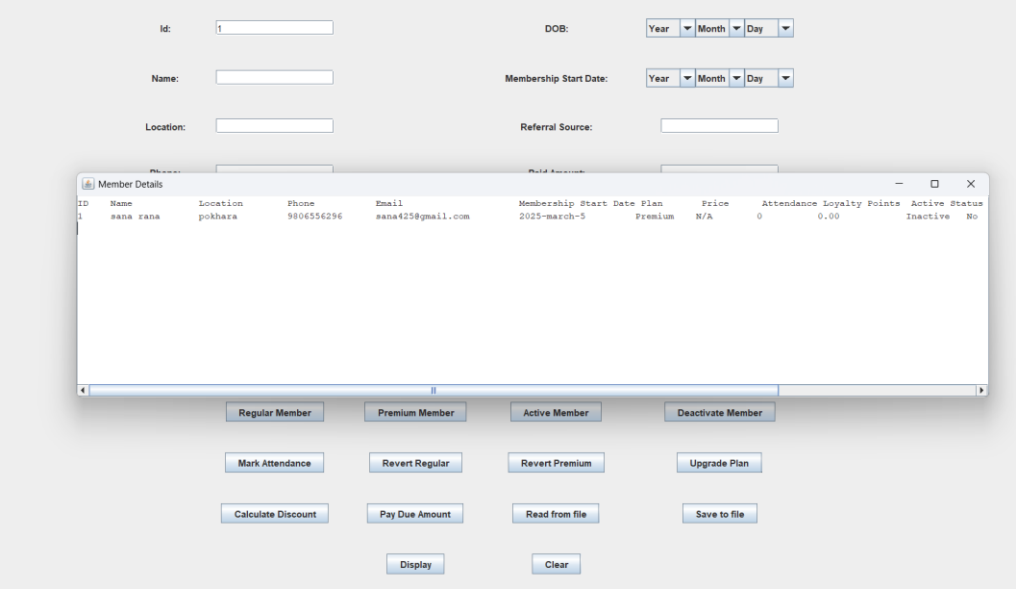

Figure 13:screenshot of read file button and displaying the details from memberDetails.txt

7.Error Detection and Correction

7.1: Compile time error or syntax error

When there is syntax error or type error then it causes a compile time error it happens before program runs.

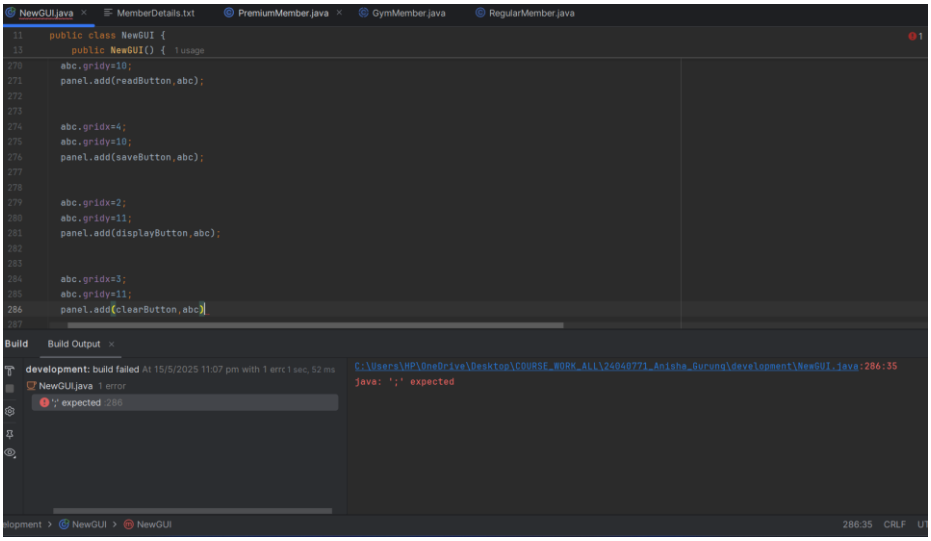

Figure 14:Syntax error due to missing of " ; "

Correction:

First, we need to identify the error where it is, then only after identifying we will try to correct. in the above code the semicolon is missing which has caused the syntax error while compile. so, now we have added the semicolon, so it has fixed the compile time error.

```
panel.add(readButton,abc);


abc.gridx=4;
abc.gridy=10;
panel.add(saveButton,abc);


abc.gridx=2;
abc.gridy=11;
panel.add(displayButton,abc);


abc.gridx=3;
abc.gridy=11;
panel.add(clearButton,abc);
```

Figure 15:correction of syntax error adding " ; "

7.2 Runtime Error

This error arises when the program is still running. When a string is converted to a numeric value and an improper format is encountered, a particular kind of runtime error called a Number Exception happens. This frequently occurs with methods like Double.parseDouble() or Integer.parseInt() when the string contains that are not numeric

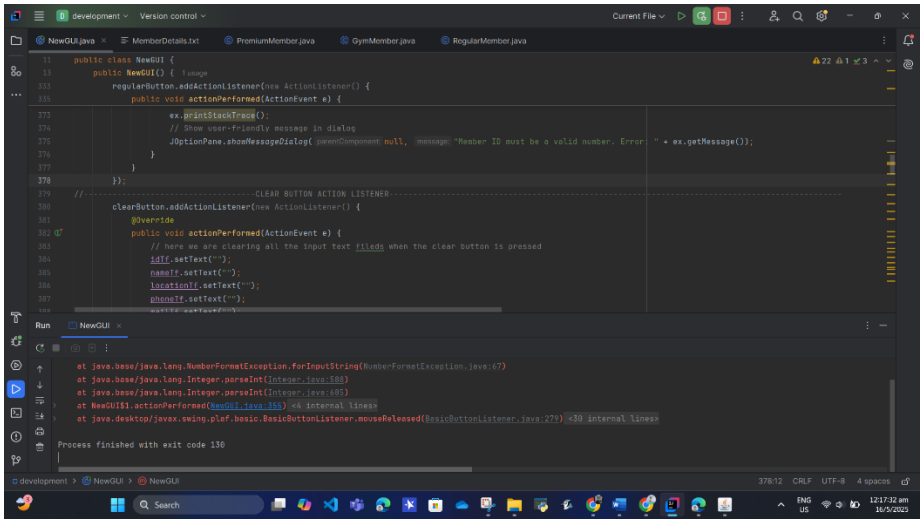

Figure 16:run time error number Format exception

Correction:

We have validated the id from text filed to convert it in integer so we can simply say that integer. paraseInt is not validating the id. so, we have converted the into the integer and used try catch for error handling and showing message to enter a valid id and Id should be a number.

```
try {
    // Collecting data from the input fields
    String idText = idTf.getText().trim();
    String name = nameTf.getText().trim();
    String location = locationTf.getText().trim();
    String phone = phoneTf.getText().trim();
    String email = mailTf.getText().trim();
    String gender = male.isSelected() ? "Male" : "Female";
    String dob = dobYearComboBox.getSelectedItem() + "-" + dobMonthComboBox.getSelectedItem() + "-" + dobDayComboBox.getSelectedItem();
    String membershipStartDate = memberYearComboBox.getSelectedItem() + "-" + memberMonthComboBox.getSelectedItem() + "-" + memberDayComboBox.getSelectedItem()
    String referralSource = referralTf.getText().trim();
    // Validate required fields
    if (idText.isEmpty() || name.isEmpty() || location.isEmpty() || phone.isEmpty() || email.isEmpty() || referralSource.isEmpty()) {
        JOptionPane.showMessageDialog( parentComponent: null,  message: "Please fill in all fields.");
        return;
    }
    int memberId = Integer.parseInt(idText); // Only this is inside try-catch
    // Check for duplicate IDs
    for (GymMember member : gymMembers) {...}
    // Create and add the RegularMember
    RegularMember regularMember = new RegularMember(memberId, name, location, phone, email, gender, dob, membershipStartDate, referralSource);
    gymMembers.add(regularMember);
    JOptionPane.showMessageDialog( parentComponent: null,  message: "Regular Member successfully added!");
} catch (NumberFormatException ex) {
    JOptionPane.showMessageDialog( parentComponent: null,  message: "Member ID must be a valid number.");
}
```

Figure 17:correction to tackle the run time error by adding try catch

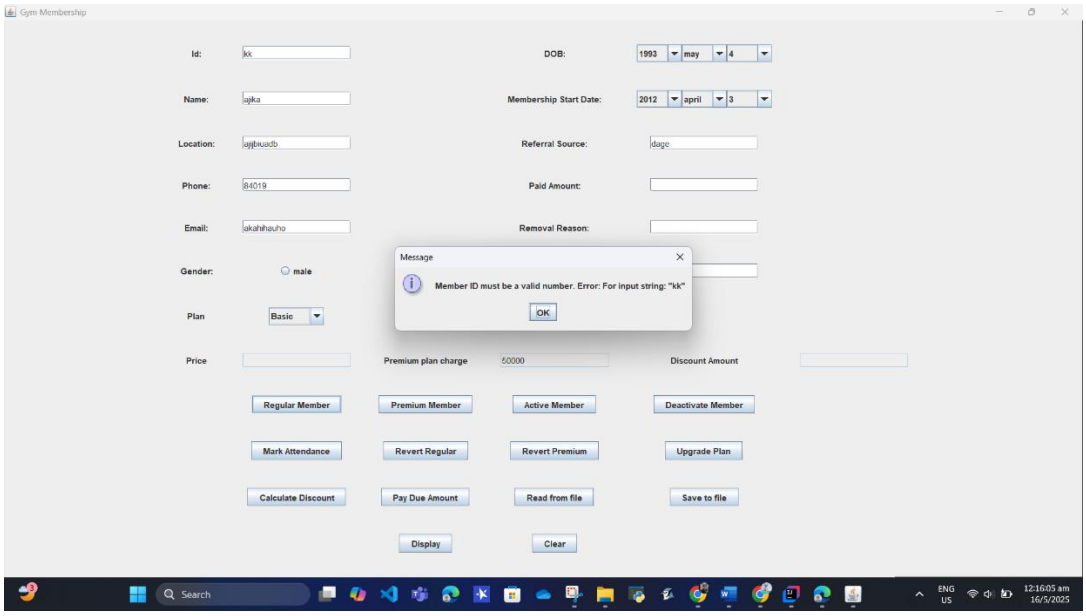

Figure 18:Popup error message to enter a valid Id

7.3 Logical error

The logical error in the code below is in such a way that even when the attendance limit is not met the user is able to upgrade the plan for regular member which is logical incorrect since I wanted to first the member needs to meet at least the attendance limit which is 30.

```
// Abstract method markAttendance
public void markAttendance() {  1 usage
    attendance++;
    loyaltyPoints += 5; // Increment loyalty points by 5
    if (attendance <= attendanceLimit) {
        isEligibleForUpgrade = true; // If attendance limit is reached, set eligible for upgrade
    }
}
```

Figure 19:Logical error in mark attendance

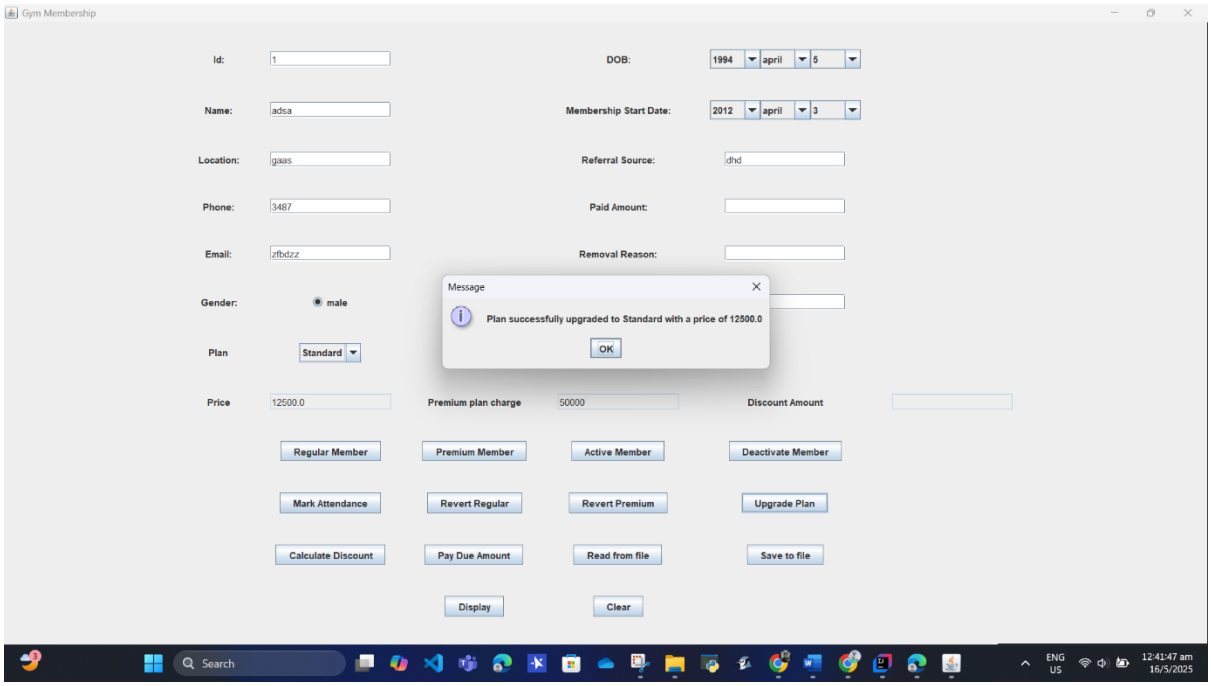

Figure 20:Even when the attendance limit is not met user able to upgrade plan

Correction:

So , in the above code there was a logical error which was caused by less than sign “ < ” because our logic was to make members only able to upgrade if the attendance limit is met which was 30 but due to this sign even when no attendance limit was met users where able to upgrade since our logic was made when members attendance is less than the limit they will be able to upgrade .so changing that sign to greater than now the logic is the attendance need to be more than or equal to attendance limit to upgrade the plan.

```
// Abstract method markAttendance
public void markAttendance() {  1 usage
    attendance++;
    loyaltyPoints += 5; // Increment loyalty points by 5
    if (attendance >= attendanceLimit) {
        isEligibleForUpgrade = true; // If attendance limit is reached, set eligible for upgrade
    }
}
```

Figure 21:Correction of logical error

8.Conclusion

As this course work requires lots of understanding about the concept of java and OOP, I had the opportunities to learn about this while working on this program. This course work is great for anyone to help his/her to fully understand how real-life applications are made. I would like to express my gratitude to our module teacher Sushil Parajuli sir for his dedication and helpful nature throughout out course work. We are grateful that our teachers have provided us with this course work so that we can have a practical knowledge about it. Not only that we have also made the graphical user interface through where a user can interact with the application in a user-friendly way. a proper Gui should be made to make sure that the user's consistency must be smooth and not sit confused because he/she cannot understand what these buttons will do. So, when need to also know the knowledge about Gui also.

Well since our project is a fully functional application for gym membership, I have got an insight knowledge about how this is worked and what should we do before starting to code like making a flowchart or writing pseudocode so that we can have an overview of our program logic how this will be working. Also adding one thing that our module teacher has given us a different assignment named independent learning where we have completed a LinkedIn course of java. there again we learned about the four pillars of java. how we can use oop to make our code proper, maintained. Also, not only that I even now have a great easiness while using the word. it was really confusing at first but as I started to work and document my project it has become easier for me. And also draw.io to draw the class diagram for our project. Many things were taught by our module teacher. What a Gui is how we can make, we learnt about the buttons, labels, text fields, radio buttons etc. also learnt about the grid bag constraint to make that buttons and Gui labels and other things to keep them align and properly according to our choice. I am thankful to my module teacher who has helped me throughout my project whenever I had any confusion regarding the java. and my friends who were supportive and understanding that they made sure to also offer help. I feel luck to have such friends and family.

Difficulties were there since this was also my first project which was functional and needed to be done in such a professional way I had encountered with numerous problems but was able to tackle it with the help of module teacher. The difficulties were

when I was unable to understand the Gui part, and there were a lot of syntax, so I was unable to remember to syntax which is not a good practice we don't need to remember it we just have to know the logic and lots of practice for it. And the object-oriented principle I had to learn that a lot because I was little bit confused in interface, inheritance part also but now with the help of module teacher and the independent learning now, I have clear picture of those topic also. For the Gui part I would like to thank my friend who had explained me how each button is first made and then how we can add action listener to it. This course work have helped me to learn so much about it and I will be always great full for this.

9. References

GeeksforGeeks, 2024. *GeeksforGeeks.* [Online]
Available at: https://www.geeksforgeeks.org/wireframing/#what-is-wireframing
[Accessed 4 March 2025].

10.Appendix of code

10.1 Appendix of regular class

```
//this is subclass or child class gym member with its own attributes and behaviours
public class RegularMember extends GymMember {
   // Private attributes for regular member
   private final int attendanceLimit = 30; // final constant which means this is the minimum limit need to match or should be more to be able to upgrade plan
   private boolean isEligibleForUpgrade;//to make sure if the member is eligible or not to have a upgrade plan
   private String removalReason;//Reason for removal
   private String referralSource;//From where they got referral
   private String plan;
   private double price;

   // Constructor this will initialize the member object
   public RegularMember(int id, String name, String location, String phone, String email,
                        String gender, String DOB, String membershipStartDate, String referralSource) {
      super(id, name, location, phone, email, gender, DOB, membershipStartDate);//this super mean this will call the constructor from the parent class
      this.referralSource = referralSource;
      this.isEligibleForUpgrade = false;// at first setting the eligiblity to false beacuse no one is eligible when first takes the membership
      this.plan = "basic";//default value of plan
      this.price = 6500;//default value of price
      this.removalReason = "";//making sure no the removal reason is empty at first
   }

   // Accessor methods which will return  attendamce,eligiblity,removal reason,referral source,plan,price
   public int getAttendanceLimit() {
      return attendanceLimit;
   }

   public boolean isEligibleForUpgrade() {
      return isEligibleForUpgrade;
   }

   public String getRemovalReason() {
      return removalReason;
   }

   public String getReferralSource() {
      return referralSource;
   }

   public String getPlan() {
```

```
    return plan;
  }

  public double getPrice() {
    return price;
  }

  public void setEligibleForUpgrade(boolean eligibleForUpgrade) {
    isEligibleForUpgrade = eligibleForUpgrade;
  }

  public void setRemovalReason(String removalReason) {
    this.removalReason = removalReason;
  }
//access method setter which will updates the attribute value fpr referra; source,plan,price
  public void setReferralSource(String referralSource) {
    this.referralSource = referralSource;
  }

  public void setPlan(String plan) {
    this.plan = plan;
  }

  public void setPrice(double price) {
    this.price = price;
  }

  // Abstract method markAttendance
  public void markAttendance() {
    attendance++;
    loyaltyPoints += 5; // Increment loyalty points by 5
    if (attendance >= attendanceLimit) {//condition is that the attendance need to be 30 or more
than 30 to be able to upgrade plan
      isEligibleForUpgrade = true; // If attendance limit is reached, set eligible for upgrade
    }
  }

  // Method to get plan price
  public double getPlanPrice(String plan) {
    switch (plan.toLowerCase()) {//since it can be case-insensitive we use tolowerCase()
      case "basic":
        return 6500;
      case "standard":
        return 12500;
      case "deluxe":
        return 18500;
      default:
        return -1; // if the invalid plan ispassed then the getplanprice() method should return -1
```

```
    }
  }

  // Method to upgrade the plan
  public String upgradePlan(String newPlan) {
    if (isEligibleForUpgrade == false) {
      return "You are not eligible for an upgrade yet.";
    } else {
      double newPrice = getPlanPrice(newPlan);
      if (newPrice == -1) { // Checking for invalid plan with if-else
        return "Invalid plan. Please select a valid plan.";
      } else {
        if (this.plan.equalsIgnoreCase(newPlan)) { // Replacing if
(!this.plan.equalsIgnoreCase(newPlan))
          return "You are already subscribed to the " + newPlan + " plan.";
        } else {
          this.plan = newPlan;
          this.price = newPrice;
          isEligibleForUpgrade = false; // Reset eligibility after upgrading
          return "Plan successfully upgraded to " + newPlan + " with a price of " + price;
        }
      }
    }
  }

  // Method to revert RegularMember details
  public void revertRegularMember(String removalReason) {
    super.resetMember(); // Call resetMember() from the superclass
    this.isEligibleForUpgrade = false;//again resetting the eligiblity to false when rvert regualar
button clicked
    this.plan = "basic";//reseting plan to basic when button reverrt is clicked
    this.price = 6500;//resetting price to 6500
    this.removalReason = removalReason;
  }

  // Display method to show RegularMember details
  public void display() {
    super.display(); // Call display() from the superclass
    System.out.println("Plan: " + plan);//
    System.out.println("Price: " + price);
    if (removalReason.equals("")) { // Replacing !removalReason.isEmpty() with if-else
      // Do nothing, no removal reason to display
    } else {
      System.out.println("Removal Reason: " + removalReason);
    }
  }
}
```

10.2 Appendix of Premium class

```
//this is also child class for gym member since it is extending the parent class
public class PremiumMember extends GymMember {
   // Private attributes  are declared within the class and not accessible directly from outside
class
   private final double premiumCharge = 50000; // it is constant
   private String personalTrainer;
   private boolean isFullPayment;
   private double paidAmount;
   private double discountAmount;

   // Constructor
   public PremiumMember(int id, String name, String location, String phone, String email,
                        String gender, String DOB, String membershipStartDate, String personalTrainer) {
      super(id, name, location, phone, email, gender, DOB, membershipStartDate);//this call the
constructor of parent class
      this.personalTrainer = personalTrainer;//setting personal trainer name
      this.isFullPayment = false;// keeping the payment to false when not paid full amount
      this.paidAmount = 0;//amount set to 0 at first
      this.discountAmount = 0;//discount amount set to 0 at first
   }

   // Accessor methods which will return all the methods
   public double getPremiumCharge() {
      return premiumCharge;
   }

   public String getPersonalTrainer() {
      return personalTrainer;
   }

   public boolean isFullPayment() {
      return isFullPayment;
   }

   public double getPaidAmount() {
      return paidAmount;
   }

   public double getDiscountAmount() {
      return discountAmount;
   }
// setter which will update the values of attributes
   public void setPersonalTrainer(String personalTrainer) {
      this.personalTrainer = personalTrainer;
   }
```

```
  public void setFullPayment(boolean fullPayment) {
    isFullPayment = fullPayment;
  }

  public void setPaidAmount(double paidAmount) {
    this.paidAmount = paidAmount;
  }

  public void setDiscountAmount(double discountAmount) {
    this.discountAmount = discountAmount;
  }

  // Implementation of the abstract method markAttendance from GymMember
  public void markAttendance() {
    // Increment attendance by 1 for PremiumMember
    attendance++;
    loyaltyPoints += 5; // Increment loyalty points by 5 as an example for PremiumMember
  }

  // Method to pay due amount
  public String payDueAmount(double amount) {
    if (isFullPayment == true) { // Check if payment is already full
      return "your payment is done.";
    }

    else {
      paidAmount += amount; // Add the paid amount

      if (paidAmount > premiumCharge) { // If total paid is greater than the total charge of
premium
        return "your payment is excees .Please check again thankyou!.";
      }
      else {
        double remainingAmount = premiumCharge - paidAmount; // Calculate remaining
amount

        if (paidAmount == premiumCharge) { // If the full payment is made
          isFullPayment = true;
          return "Payment successful!  Premium charge of " + premiumCharge + " has been
paid.";

        } else { // If payment is not full
          return "Payment successful! Remaining amount: " + remainingAmount;
        }
      }
    }
  }
```

```
  // Method to calculate discount
  public void calculateDiscount() {
    if (isFullPayment == true) { // If full payment is made
      discountAmount = 0.1 * premiumCharge; // 10% discount on premium charge
      System.out.println("Discount of " + discountAmount + " has been applied to your
premium charge.");
    } else { // If full payment is not made
      discountAmount = 0; // No discount available if payment is not full
      System.out.println("No discount available as payment is not full. please consider to
pay!!");
    }
  }

  // Method to revert PremiumMember details
  public void revertPremiumMember() {
    super.resetMember(); // Call resetMember() from the superclass
    this.personalTrainer = ""; // Reset personalTrainer to empty
    this.isFullPayment = false;//resetting full payment to false
    this.paidAmount = 0;// reset amount to 0
    this.discountAmount = 0; //reset discount amount to 0
  }

  // Display method to show PremiumMember details
  public void display() {
    super.display(); // Call display() from the superclass which is parent class
    System.out.println("Personal Trainer: " + personalTrainer);
    System.out.println("Paid Amount: " + paidAmount);
    System.out.println("Full Payment Status: " + isFullPayment);

    // Calculate and display remaining amount if payment is not full
    double remainingAmount = premiumCharge - paidAmount;
    if (isFullPayment == false) { // If payment is not full
      System.out.println("Remaining Amount to be Paid: " + remainingAmount);
    }

    // Display discount amount if full payment is done
    if (isFullPayment == true) { // If full payment is done
      System.out.println("Discount Amount: " + discountAmount);
    }
  }
}
```

10.3:Appendix of NewGUI Class

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
import java.util.ArrayList;//imports arraylist for storing gymm members
import java.io.PrintWriter;
import java.io.FileNotFoundException;
import java.io.BufferedReader;
import java.io.FileReader;
public class NewGUI {//this is the gui where all the buttons text fields are kept
   ArrayList<GymMember> gymMembers = new ArrayList<>();
   public NewGUI() {// this is the constructor
      JFrame frame = new JFrame("Gym Application");// frame name gym Application
      JPanel panel = new JPanel(new GridBagLayout());//panel with grib bag lay out
//radio button if we have to select one from option
      JRadioButton male = new JRadioButton("male");
      JRadioButton female = new JRadioButton("female");
      ButtonGroup group = new ButtonGroup();
      group.add(male);
      group.add(female);
//grouping the buttons so it acts as one and when one button is selected another unchoose
itself
      JLabel id = new JLabel("Id:");// this are labels for id ,name locarion and so on  it is kept at the
side of the textfield to make user easy so
                              //they will know in which text field the data need to be entered
      JLabel name = new JLabel("Name:");
      JLabel location = new JLabel("Location:");
      JLabel phone = new JLabel("Phone:");
      JLabel mail = new JLabel("Email:");
      JLabel gender = new JLabel("Gender:");
      JLabel referralSource = new JLabel("Referral Source:");
      JLabel paidAmount = new JLabel("Paid Amount:");
      JLabel removalReason = new JLabel("Removal Reason:");
      JLabel trainerName = new JLabel("Trainer's Name:");
      JLabel plan=new JLabel("Plan");
      JLabel price=new JLabel("Price");
      JLabel premiumPlanCharge=new JLabel("Premium plan charge");
      JLabel discountAmount=new JLabel("Discount Amount");
//combo box for dob
      JLabel dob = new JLabel("DOB:");
      JPanel dobComboBoxPanel = new JPanel(new GridLayout(1, 3));
      JComboBox<String> dobYearComboBox = new JComboBox<>(new String[]{"Year", "1991",
"1992", "1993", "1994", "1995", "1996", "1997", "1998", "1999", "2000", "2001", "2002", "2003",
"2004", "2005", "2006", "2007", "2008"});
      JComboBox<String> dobMonthComboBox = new JComboBox<>(new String[]{"Month", "jan",
"feb", "march", "april", "may", "june", "july", "aug", "sept", "oct", "nov", "dec"});
      JComboBox<String> dobDayComboBox = new JComboBox<>(new String[]{"Day", "1", "2",
"3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21",
"22", "23", "24", "25", "26", "27", "28", "29", "30", "31"});

      JLabel membershipStartDate = new JLabel("Membership Start Date:");
      JPanel memberComboBoxPanel = new JPanel(new GridLayout(1, 3));
```

```
    JComboBox<String> memberYearComboBox = new JComboBox<>(new String[]{"Year",
"2010", "2011", "2012", "2013", "2014", "2015", "2016", "2017", "2018", "2019", "2020", "2021",
"2022", "2023", "2024", "2025"});
    JComboBox<String> memberMonthComboBox = new JComboBox<>(new String[]{"Month",
"jan", "feb", "march", "april", "may", "june", "july", "aug", "sept", "oct", "nov", "dec"});
    JComboBox<String> memberDayComboBox = new JComboBox<>(new String[]{"Day", "1",
"2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20",
"21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31"});
//TEXTFIELDS to enter data and at the side of this text fields the necessary labels are kept to
make easy to recognize
    JTextField idTf = new JTextField(15);
    JTextField nameTf = new JTextField(15);
    JTextField locationTf = new JTextField(15);
    JTextField phoneTf = new JTextField(15);
    JTextField mailTf = new JTextField(15);
    JTextField referralTf = new JTextField(15);
    JTextField paidTf = new JTextField(15);
    JTextField removalTf = new JTextField(15);
    JTextField trainerNameTf = new JTextField(15);
    JTextField priceTf=new JTextField(15);
    priceTf.setEditable(false);
    JTextField premiumChargeTf=new JTextField(15);
    premiumChargeTf.setEditable(false);
    premiumChargeTf.setText("50000");
    JTextField discountTF=new JTextField(15);
    discountTF.setEditable(false);

    //BUTTON FOR THE ACTION WHICH WILL PERFORM ACCORDING TO THEIR METHODS
    JButton regularButton = new JButton("Regular Member");
    JButton premiumButton = new JButton("Premium Member");
    JButton activeButton = new JButton("Active Member");
    JButton deactivateButton = new JButton("Deactivate Member");
    JButton attendanceButton = new JButton("Mark Attendance");
    JButton upgradeButton = new JButton("Upgrade Plan");
    JButton calculateButton = new JButton("Calculate Discount");
    JButton revertRegular = new JButton("Revert Regular");
    JButton revertPremium=new JButton("Revert Premium");
    JButton paymentButton = new JButton("Pay Due Amount");
    JButton readButton = new JButton("Read from file");
    JButton saveButton = new JButton("Save to file");
    JButton displayButton = new JButton("Display");
    JButton clearButton = new JButton("Clear");
    JComboBox planComboBox = new JComboBox<>(new String[]{"Basic", "Standard",
"Deluxe"});
//Using grid bag constraibts and has a variable name abc
    GridBagConstraints abc=new GridBagConstraints();
    abc.insets=new Insets(20,20,20,20);// this is for the padding the components
//x is for x-axis and y is for y-axis and adding that componet to the panel
```

```
abc.gridx=0;
abc.gridy=0;
panel.add(id,abc);

abc.gridx=1;
abc.gridy=0;
panel.add(idTf,abc);

abc.gridx=0;
abc.gridy=1;
panel.add(name,abc);

abc.gridx=1;
abc.gridy=1;
panel.add(nameTf,abc);

abc.gridx=0;
abc.gridy=2;
panel.add(location,abc);

abc.gridx=1;
abc.gridy=2;
panel.add(locationTf,abc);

abc.gridx=0;
abc.gridy=3;
panel.add(phone,abc);

abc.gridx=1;
abc.gridy=3;
panel.add(phoneTf,abc);

abc.gridx=0;
abc.gridy=4;
panel.add(mail,abc);

abc.gridx=1;
abc.gridy=4;
panel.add(mailTf,abc);

abc.gridx=0;
abc.gridy=5;
panel.add(gender,abc);

abc.gridx=1;
abc.gridy=5;
panel.add(male,abc);
```

```
abc.gridx=2;
abc.gridy=5;
panel.add(female,abc);

abc.gridx=3;
abc.gridy=0;
panel.add(dob,abc);

abc.gridx=4;
abc.gridy=0;
dobComboBoxPanel.add(dobYearComboBox);
dobComboBoxPanel.add(dobMonthComboBox);
dobComboBoxPanel.add(dobDayComboBox);
panel.add(dobComboBoxPanel,abc);

abc.gridx=3;
abc.gridy=1;
panel.add(membershipStartDate,abc);

abc.gridx=4;
abc.gridy=1;

memberComboBoxPanel.add(memberYearComboBox);
memberComboBoxPanel.add(memberMonthComboBox);
memberComboBoxPanel.add(memberDayComboBox);
panel.add(memberComboBoxPanel,abc);

abc.gridx=3;
abc.gridy=2;
panel.add(referralSource,abc);

abc.gridx=4;
abc.gridy=2;
panel.add(referralTf,abc);

abc.gridx=3;
abc.gridy=3;
panel.add(paidAmount,abc);

abc.gridx=4;
abc.gridy=3;
panel.add(paidTf,abc);

abc.gridx=3;
abc.gridy=4;
panel.add(removalReason,abc);

abc.gridx=4;
```

```
abc.gridy=4;
panel.add(removalTf,abc);


abc.gridx=3;
abc.gridy=5;
panel.add(trainerName,abc);


abc.gridx=4;
abc.gridy=5;
panel.add(trainerNameTf,abc);

abc.gridx=0;
abc.gridy=6;
panel.add(plan,abc);

abc.gridx=1;
abc.gridy=6;
panel.add(planComboBox,abc);

abc.gridx=0;
abc.gridy=7;
panel.add(price,abc);

abc.gridx=1;
abc.gridy=7;
panel.add(priceTf,abc);

abc.gridx=2;
abc.gridy=7;
panel.add(premiumPlanCharge,abc);

abc.gridx=3;
abc.gridy=7;
panel.add(premiumChargeTf,abc);

abc.gridx=4;
abc.gridy=7;
panel.add(discountAmount,abc);

abc.gridx=5;
abc.gridy=7;
panel.add(discountTF,abc);

abc.gridx=1;
abc.gridy=8;
panel.add(regularButton,abc);
```

```
abc.gridx=2;
abc.gridy=8;
panel.add(premiumButton,abc);

abc.gridx=3;
abc.gridy=8;
panel.add(activeButton,abc);

abc.gridx=4;
abc.gridy=8;
panel.add(deactivateButton,abc);

abc.gridx=1;
abc.gridy=9;
panel.add(attendanceButton,abc);

abc.gridx=2;
abc.gridy=9;
panel.add(revertRegular,abc);

abc.gridx=3;
abc.gridy=9;
panel.add(revertPremium,abc);

abc.gridx=4;
abc.gridy=9;
panel.add(upgradeButton,abc);

abc.gridx=1;
abc.gridy=10;
panel.add(calculateButton,abc);


abc.gridx=2;
abc.gridy=10;
panel.add(paymentButton,abc);


abc.gridx=3;
abc.gridy=10;
panel.add(readButton,abc);


abc.gridx=4;
abc.gridy=10;
panel.add(saveButton,abc);
```

```
    abc.gridx=2;
    abc.gridy=11;
    panel.add(displayButton,abc);


    abc.gridx=3;
    abc.gridy=11;
    panel.add(clearButton,abc);

    //-----------------------------------------------ACTION----LISTENER-----------------------------------------
------------/
    regularButton.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        try {
          // Collecting data from the input fields
          String idText = idTf.getText().trim();
          String name = nameTf.getText().trim();
          String location = locationTf.getText().trim();
          String phone = phoneTf.getText().trim();
          String email = mailTf.getText().trim();
          String gender = male.isSelected() ? "Male" : "Female";
          String dob = dobYearComboBox.getSelectedItem() + "-" +
dobMonthComboBox.getSelectedItem() + "-" + dobDayComboBox.getSelectedItem();
          String membershipStartDate = memberYearComboBox.getSelectedItem() + "-" +
memberMonthComboBox.getSelectedItem() + "-" + memberDayComboBox.getSelectedItem();
          String referralSource = referralTf.getText().trim();
          // Validate required fields is enpty then so the message
          if (idText.isEmpty() || name.isEmpty() || location.isEmpty() || phone.isEmpty() ||
email.isEmpty() || referralSource.isEmpty()) {
            JOptionPane.showMessageDialog(null, "Please fill in all fields which are empty.");
            return;
          }
          int memberId = Integer.parseInt(idText); // Only this is inside try-catch and this
converts the string into integer beacuse the id is inputed as string when enered and onverted it
into the integer

          // Check for duplicate IDs
          for (GymMember member : gymMembers) {
            if (member.getId() == memberId) {
              JOptionPane.showMessageDialog(null, "This ID is already taken. Please choose a
different ID.");
              return;
            }
          }
          // Create and add the RegularMember
          RegularMember regularMember = new RegularMember(memberId, name, location,
```

```
phone, email, gender, dob, membershipStartDate, referralSource);
            gymMembers.add(regularMember);
            JOptionPane.showMessageDialog(null, "Regular Member  is successfully added!");
          }
          catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(null, "Sorry !!!Member ID must be a valid
number.");
          }
        }
      });

//-----------------------------------CLEAR BUTTON ACTION LISTENER-----------------------------------
-------------------------------------------------------
      clearButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
          // here we are clearing all the input text fileds when the clear button is pressed
          idTf.setText("");
          nameTf.setText("");
          locationTf.setText("");
          phoneTf.setText("");
          mailTf.setText("");
          referralTf.setText("");
          removalTf.setText("");
          trainerNameTf.setText("");
          priceTf.setText("");
          discountTF.setText("");
          paidTf.setText("");

          // Resetting the  combo boxes to the default option of index 0 so they still show year
month and day
          dobYearComboBox.setSelectedIndex(0);
          dobMonthComboBox.setSelectedIndex(0);
          dobDayComboBox.setSelectedIndex(0);

          memberYearComboBox.setSelectedIndex(0);
          memberMonthComboBox.setSelectedIndex(0);
          memberDayComboBox.setSelectedIndex(0);

          // Unchecking gender radio buttons
          group.clearSelection();

          // here resetting the plan combo box to default value of index 0
          planComboBox.setSelectedIndex(0);
          JOptionPane.showMessageDialog(null, "All fields have been cleared.");
        }
      });
```

```
    //--------------PREMIUM BUTTON ACTION LISTENER-----------------------------------------------------
-------------------------------------------------
    premiumButton.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        try {
          // Collecting the input data
          String idText = idTf.getText().trim();
          String name = nameTf.getText().trim();
          String location = locationTf.getText().trim();
          String phone = phoneTf.getText().trim();
          String email = mailTf.getText().trim();
          String gender = male.isSelected() ? "Male" : (female.isSelected() ? "Female" : "");
          String dob = dobYearComboBox.getSelectedItem() + "-" +
dobMonthComboBox.getSelectedItem() + "-" + dobDayComboBox.getSelectedItem();
          String membershipStartDate = memberYearComboBox.getSelectedItem() + "-" +
memberMonthComboBox.getSelectedItem() + "-" + memberDayComboBox.getSelectedItem();
          String trainerName = trainerNameTf.getText().trim();

          // Validate required fields if empty to the message box
          if (idText.isEmpty() || name.isEmpty() || location.isEmpty() || phone.isEmpty() ||
              email.isEmpty() || gender.isEmpty() || trainerName.isEmpty()) {
            JOptionPane.showMessageDialog(null, "Sorry !!!Please fill in all required  fields.");
            return;
          }

          // Validate ID
          int memberId = Integer.parseInt(idText);//again convert the id entered into the integer

          // Check for duplicate ID if the id is found already in the system then sho the message
          for (GymMember member : gymMembers) {
            if (member.getId() == memberId) {
              JOptionPane.showMessageDialog(null, "Sorry!!!This ID is already taken. Please
choose a new ID.");
              return;
            }
          }
          // Create and add PremiumMember to the gym memebers list
          PremiumMember premiumMember = new PremiumMember(memberId, name,
location, phone, email, gender, dob, membershipStartDate, trainerName);
          gymMembers.add(premiumMember);

          JOptionPane.showMessageDialog(null, "Premium Member successfully added!");

        } catch (NumberFormatException ex) {
          JOptionPane.showMessageDialog(null, "Member ID must be a valid number.");
        }
```

```
        }
    });
    //------------------------------ACTIVE MEMBERSHIP BUTTON ACTION LISTENER----------------------
-------------------------------------------------------------------
    activeButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            // 1. Get the Member ID from the input field
            String idText = idTf.getText().trim();  // Get the text from the ID field

            // 2. Check if the ID field is empty
            if (idText.isEmpty()) {
                JOptionPane.showMessageDialog(null, "Please enter the Member ID to activate.");
                return;
            }

            try {
                // 3. Convert the entered text to an integer (Member ID)
                int memberId = Integer.parseInt(idText);  // Convert the input into an integer
                boolean found = false;  // to check if the ID was found

                // 4. Loop through the gymMembers list to find a match
                for (GymMember member : gymMembers) {
                    if (member.getId() == memberId) {
                        // . If the Member ID matches, activate the membership
                        member.activateMembership();  // Call the method from GymMember
                        JOptionPane.showMessageDialog(null, "Membership activated for Member ID: " +
memberId+ "name:"+member.getName());// to when active button or deactive button is
pressed we are using getNAme()*because whe are displaying the member which are already
existing not getTExt()
                        found = true;  // Set the flag to true indicating the member was found
                        break;  // Exit the loop
                    }
                }

                // 6. If the Member ID was not found, show an error message
                if (!found) {
                    JOptionPane.showMessageDialog(null, "sorry!! there is no member with this ID : " +
memberId);
                }

            } catch (NumberFormatException ex) {
                // 7. If the entered ID is not a valid number, show an error message
                JOptionPane.showMessageDialog(null, "Sorry,Member ID must be a valid number.");
            }
        }
    });
//---------------------------------ACTION LISTENER OF DEACTIVATE BUTTON----------------------------
```

```
-----------------------------------------
    deactivateButton.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        String idText = idTf.getText().trim();

        // Check if ID is entered if empty show the message below
        if (idText.isEmpty()) {
          JOptionPane.showMessageDialog(null, "Sorry!!!!Please enter  ID to deactivate.");
          return;
        }

        try {
          // 3. Convert the entered text to an integer (Member ID)
          int memberId = Integer.parseInt(idText);
          boolean found = false;

          for (GymMember member : gymMembers) {
            if (member.getId() == memberId) {
              member.deactivateMembership();  // Call method from GymMember class
              JOptionPane.showMessageDialog(null, "Membership deactivated for Member ID:
" + memberId + ", Name: " + member.getName());
              found = true;
              break;
            }
          }

          if (!found) {
            JOptionPane.showMessageDialog(null, "No member found with ID: " + memberId);
          }

        } catch (NumberFormatException ex) {
          JOptionPane.showMessageDialog(null, "Invalid Member ID. Please enter a valid
number.");
        }
      }
    });
    //---------------------------------- REVERT REGULAR BUTTON----------------------------------------
------
    revertRegular.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        String idText = idTf.getText().trim();  // Use the ID field for member ID
         String removalReason =removalTf.getText().trim();
        if (idText.isEmpty() || removalReason.isEmpty()) {
          JOptionPane.showMessageDialog(null, "Please enter Member ID. And Also write your
removal reason");
          return;
```

```
            }
            try {
                int memberId = Integer.parseInt(idText); // Convert the entered text to an integer
(Member ID)
                boolean found = false;

                for (GymMember member : gymMembers) {
                    if (member.getId() == memberId && member instanceof RegularMember) {
                        ((RegularMember) member).revertRegularMember(removalReason);
                        JOptionPane.showMessageDialog(null, " Regular Member reverted: " +
member.getName()+"Reason of Removal:"+removalReason);
                        found = true;
                        break;
                    }
                }
                if (!found) {
                    JOptionPane.showMessageDialog(null, "Sorry!!! No id is found : " + memberId);
                }

            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(null, " Sorry !!!Please enter a valid  ID.");
            }
        }
    });
//----------------------------------REVERT PREMIUM BUTTON-----------------------------------------------
-------------------------------------------------
    revertPremium.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String idText = idTf.getText().trim();  // Use the ID field for member ID


            if (idText.isEmpty()) {
                JOptionPane.showMessageDialog(null, "Sorry!!!Please enter the required Member ID
");
                return;
            }
            try {
                //Convert the entered text to an integer (Member ID)
                int memberId = Integer.parseInt(idText);
                boolean found = false;

                for (GymMember member : gymMembers) {
                    if (member.getId() == memberId && member instanceof PremiumMember) {
                        ((PremiumMember) member).revertPremiumMember();
                        JOptionPane.showMessageDialog(null, "Premium Member reverted successfully
for: " + member.getName());
                        found = true;
```

```
                break;
              }
            }

            if (!found) {
              JOptionPane.showMessageDialog(null, "id cannot be found: " + memberId);
            }

          } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(null, "Please enter a valid number.");
          }
        }
      });
//-----------------------------------ATTENDANCE BUTTON-----------------------------------------------------
-------------------------------------------------------------
      attendanceButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
          try {
            // Get the text from the ID input field and trim any spaces
            String inputText = idTf.getText().trim();

            //  Convert the input text to an integer (Member ID)
            int enteredId = Integer.parseInt(inputText);

            // Initialize a variable to hold the matching GymMember
            GymMember foundMember = null;
            //gymMember is class foundMember is variable and null means that is it right now
empty and later it will find the id while looping

            // Step 4: Loop through the gymMembers list to find a member with the entered ID
            for (GymMember member : gymMembers) {
              if (member.getId() == enteredId) {
                foundMember = member; // Store the found member
                break; // Exit the loop since we found the member
              }
            }

            // Step 5: If a matching member was found
            if (foundMember != null) {
              // Call the member's markAttendance() method
              foundMember.markAttendance();

              // Show a success message with updated attendance and loyalty points
              JOptionPane.showMessageDialog(null, " Attendance marked for: " +
foundMember.getName() +
                      "\n Attendance count: " + foundMember.getAttendance() +
                      "\n Loyalty Points: " + foundMember.getLoyaltyPoints());
```

```
            }
            /*
          foundMember.getName() return the member's name which is in gym member class
          */
            else {
              // If no member with the entered ID was found, show a basic error message
              JOptionPane.showMessageDialog(null, " id is not available");
            }
          } catch (NumberFormatException ex) {
            // If the entered ID is not a valid number, show a simple text message (no warning
icon)
            JOptionPane.showMessageDialog(null, "Please enter a valid member id.");
          }
        }
      });
      //------------------------------PAY DUE AMOUNT-----------------------------------------------------------------
----------------
      paymentButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
          String inputId=idTf.getText().trim();
          String paymentInput=paidTf.getText().trim();
          if(inputId.isEmpty() || paymentInput.isEmpty()){
            JOptionPane.showMessageDialog(null,"sorry you have not entered the ID  or Amount .
Please enter valid ID and amount ");
            return;
          }
          //  Convert the input text to an integer (Member ID)
          int idEnter=Integer.parseInt(inputId);
          double amountEnter=Double.parseDouble(paymentInput);

          GymMember searchId=null;
          for(GymMember memberPay:gymMembers){
            if(memberPay.getId()==idEnter){
              searchId=memberPay;
              break;
            }
          }

          if(searchId!=null){
            if(searchId instanceof PremiumMember){
              PremiumMember member=(PremiumMember) searchId;
              String paymentMessage=member.payDueAmount(amountEnter);
              JOptionPane.showMessageDialog(null,paymentMessage);
            }
            else {
              JOptionPane.showMessageDialog(null,"Sorry ,the entered id is not a premium
member.");
```

```
            }
          }
          else {
            JOptionPane.showMessageDialog(null,"sorry we all unable to find the id");
          }
        }
      });

    //----------------------------------------Discount--Amount--Button----------------------------------
-------------------------------------------------------
          calculateButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
              String idEnter=idTf.getText().trim();
              if (idEnter.isEmpty()) {
                JOptionPane.showMessageDialog(null, "Sorry,You have not entered the ID. Please
enter ID first");
                return;
              }
              //  Convert the input text to an integer (Member ID)

              int idInput=Integer.parseInt(idEnter);

              GymMember searchId=null;
              for(GymMember calculate :gymMembers){
                if(calculate.getId()==idInput){
                  searchId=calculate;
                  break;
                }
              }

              if(searchId!=null){
                if(searchId instanceof PremiumMember){
                  PremiumMember discount=(PremiumMember) searchId;

                  if(discount.isActiveStatus()){
                    discount.calculateDiscount();

                    double discountAmount=discount.getDiscountAmount();
                    discountTF.setText(String.valueOf(discountAmount));
                  }
                  else {
                    JOptionPane.showMessageDialog(null,"sorry,this entered id is not active");
                  }
                }
              }
              else{
                JOptionPane.showMessageDialog(null,"Sorry,we are unable to the find the id");
```

```
            }

          }
        });


 //-----------------------------------------READ TO FILE-----------------------------------------------------------
-----------------------------------------
    readButton.addActionListener(e -> {
      try {
        // Create new frame
        JFrame readFrame = new JFrame("Member Details");//creates new frame for the read
        readFrame.setSize(600, 300);
        readFrame.setLocationRelativeTo(null);//it makes to open the frame in center
        readFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);//close window
without exiting app

        // Create text area to put all the details of members
        JTextArea textArea = new JTextArea();
        textArea.setFont(new Font("Monospaced", Font.PLAIN, 12));//font is monospaced,size
12
        textArea.setEditable(false);//only able to read

        // Read file
        BufferedReader reader = new BufferedReader(new FileReader("MemberDetails.txt"));
        // this will open the member details.txt file for the reading
        String line;
        while ((line = reader.readLine()) != null) {
          textArea.append(line + "\n");//this will add the every line to the text area
        }
        reader.close();//this will close the file

        // Add text area to scroll pane
        readFrame.add(new JScrollPane(textArea));//adding the readframe to the scroll plane
        readFrame.setVisible(true);//setting the frame to visible
      } catch (Exception ex) {
        JOptionPane.showMessageDialog(null, "Error: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
      }
    });
    //-------------DISPLAY ACTION LISTENER-----------------------------------------------------------------------
-----------------------------------------------------
    displayButton.addActionListener(e -> {
      // before displaying we need to check if the gym array list is empty or not
      if (gymMembers.isEmpty()) {
        JOptionPane.showMessageDialog(null, "Sorry!!!No members to display!", "Warning",
JOptionPane.WARNING_MESSAGE);
        return;
```

```
        }

        // Create new frame to display
        JFrame displayFrame = new JFrame("Member Details");//name of frame is member details
        displayFrame.setSize(600, 300);
        displayFrame.setLocationRelativeTo(null);//this hepls to open the window in the center
        displayFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);// this makes
sure that the frame closes with out eixting the app

        // Create text area for members details.
        JTextArea textArea = new JTextArea();
        textArea.setFont(new Font("Monospaced", Font.PLAIN, 12));
        textArea.setEditable(false);

        // creates text area for members details and adds header
        textArea.append(String.format("%-5s %-15s %-15s %-15s %-25s %-20s %-10s %-10s %-
10s %-15s %-10s %-15s %-15s %-15s\n",
            "ID", "Name", "Location", "Phone", "Email", "Membership Start Date",
            "Plan", "Price", "Attendance", "Loyalty Points", "Active Status",
            "Full Payment", "Discount Amount", "Net Amount Paid"));

        // Add member details  to the  gymmembers
        for (GymMember member : gymMembers) {
          if (member instanceof RegularMember rm) {
            //adds the text area  to the regular members details
            textArea.append(String.format("%-5d %-15s %-15s %-15s %-25s %-20s %-10s %-
10.2f %-10d %-15.2f %-10s %-15s %-15s %-15s\n",
                member.getId(), member.getName(), member.getLocation(), member.getPhone(),
                member.getEmail(), member.getMembershipStartDate(),
                rm.getPlan(), rm.getPrice(), member.getAttendance(),
                member.getLoyaltyPoints(), member.isActiveStatus() ? "Active" : "Inactive",
                "N/A", "N/A", "N/A"));
            //add the text area to the premium members details
          } else if (member instanceof PremiumMember pm) {
            textArea.append(String.format("%-5d %-15s %-15s %-15s %-25s %-20s %-10s %-10s
%-10d %-15.2f %-10s %-15s %-15.2f %-15.2f\n",
                member.getId(), member.getName(), member.getLocation(), member.getPhone(),
                member.getEmail(), member.getMembershipStartDate(),
                "Premium", "N/A", member.getAttendance(),
                member.getLoyaltyPoints(), member.isActiveStatus() ? "Active" : "Inactive",
                pm.isFullPayment() ? "Yes" : "No", pm.getDiscountAmount(),
pm.getPaidAmount()));
          }
        }
        // Add text area to scroll pane
        displayFrame.add(new JScrollPane(textArea));
        displayFrame.setVisible(true);
    });
```

```
    //-------------------------------------------------------------------------------------------------------------------
-----------------------------------
    frame.add(panel);//adding the panel to the frame
        frame.setSize(1000, 800);
        frame.setLocationRelativeTo(null);//adding this makes sures that the frame centers
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);//setting the frame visible

      }

      public static void main(String[] args) {
        new NewGUI();
      }
    }
```